MOTION IMAGERY STANDARDS BOARD **Recommended Practice** **Real-Time Protocol for Full Motion Video**	**MISB RP 0804.3** **30 September 2010**

1 Scope

This Recommended Practice (RP) documents the standards profile for packaging and delivering Full Motion Video (FMV as defined in MISP 4.5 and later) data over the Real-Time Protocol (RTP). This RP provides direction on the packetization and streaming of video and metadata using RTP to support diverse IP based networks.

The scope of this RP is limited to delivery of Full Motion Video products and is not intended to replace any other approved standards for other uses; rather it is intended to complement those standards.

2 References

2.1 Normative References

[1] ISMA 2.0, *Internet Streaming Media Alliance Implementation Specification,* April 2005

[2] RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*, July 2003

[3] RFC 3551, *RTP Profile for Audio and Video Conferences with Minimal Control*, July 2003

[4] ISO/IEC 13818-2: 2000, Information Technology – Generic coding of moving pictures and associated audio information: Video

[5] ITU-T Rec. H.264 *Advanced Video Coding for Generic Audio Visual Services*, 03/2009. (ISO/IEC 14496-10:2009)

[6] MISB EG 0802, *H.264/AVC Coding and Multiplexing*, May 2009

[7] MISB EG 0104.5, *Predator UAV Basic Universal Metadata Set*, May 2008

[8] MISB STANDARD 0601.4, *UAS Datalink Local Metadata Set*, March 2010

[9] MISB STANDARD 0902.1, MISB Minimum Metadata Set, June 2010

[10] MISB STANDARD 0604.1, *Time Stamping Compressed Motion Imagery,* September 2009

[11] SMPTE 336M-2007, *Data Encoding Protocol Using Key-Length-Value*

[12] MISB RP 0701 *Common Metadata System: Structure,* August 2007

[13] RFC 2250, *RTP Payload Format for MPEG1/MPEG2 Video*, January 1998

[14] DVB IP Phase 1 handbook, ETSI TS 102 034, *Digital Video Broadcasting (DVB); Transport of MPEG-2 Based DVB Services over IP Based Networks*, March 2005

[15] RFC 3984, *RTP Payload Format for H.264 Video*, February 2005

[16] RFC 2326, *Real Time Streaming Protocol (RTSP)*, April 1998

[17] RFC 2327, *SDP: Session Description Protocol*, April 1998

[18] SMPTE 2022-2-2007, *Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks*

[19] Pro-MPEG Code of Practice #3 release 2 July 2004, *Transmission of Professional MPEG-2 Transport Streams over IP Networks*

[20] draft-ietf-avt-rtp-klv-00.txt, *RTP Payload Format for SMPTE 336M Encoded Data*

2.2 Informative References

[21] SMPTE 335M-2001, *Metadata Dictionary Structure*

[22] SMPTE RP210.8-2004, *Metadata Dictionary*

[23] SMPTE RP210.7-2003, *Metadata Dictionary*

[24] SMPTE RP210.3-2001, *Metadata Dictionary* (DRAFT)

[25] MISP 5.1, *Motion Imagery Standards Profile*, May 2008

[26] MISB RP 0101, *Use of MPEG-2 Systems Streams in Digital Motion Imagery Systems*

[27] MISB RP 0103.1, *Timing Reconciliation Metadata Set for Digital Motion Imagery*, 11 October 2001

[28] MISB RP 0107, *Bit and Byte Order for Metadata in Motion Imagery Files and Streams,* 11 October 2001

[29] MISB TRM-07A, *Low Bandwidth Motion Imagery – Technologies*, November 2007

[30] *IP Streaming of MPEG-4: Native RTP vs. MPEG-2 Transport Stream,* envivio, October 2005

[31] RTP - Audio and Video for the Internet, Colin Perkins, November 2006

3 Acronyms

DTS	Decode Time Stamp
FMV	Full Motion Video
H.264/AVC	MPEG-4 Part 10 Video Codec
IP	Internet Protocol
ISMA	Internet Streaming Media Alliance

MPEG2 TS	MPEG2 Transport Stream
PTS	Presentation Time Stamp
RTP	Real Time Protocol
RTP/AVP	IETF's RTP using Audio/Video profile carried over UDP
RTCP	Real Time Control Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator

4 Introduction

This RP defines the standards profile used to provide FMV to users via RTP over IP (Internet Protocol) based networks. The scope of what this RP is attempting to provide is very broad, and given the wide variety of infrastructure, client device, user requirements, and other considerations, it does not attempt to specifically address all possible configurations. Rather, this RP focuses on the standards that provide broad flexibility, and reference implementation guidance for RTP to determine how to best apply it for specific needs.

Unlike MPEG2 Transport Stream (TS), RTP does not support the multiplexing of media streams together within one unified package; each media stream is carried as a separate RTP stream. Therefore, each RTP media stream must contain sufficient timing information for synchronizing related streams at the receiver. Mapping a video elementary stream, such as an encoded MPEG2 or H.264/AVC elementary stream directly to RTP is called *native RTP carriage*. MPEG-2 transport stream can also be carried over RTP. MPEG-2 TS over RTP requires additional header overhead. Because both types of carriage are found in practice they will be included here for completeness.

Following a brief overview of the features and relations to existing standards of RTP, section 6 describes native RTP carriage of MPEG2 and H.264/AVC, while section 7 describes carriage of MPEG2 transport stream media over RTP. In section 8, guidelines for mapping the individual media components of a MPEG2 transport stream to individual RTP streams are given. Streaming technologies to support timed RTP media are discussed in section 10.

Appendix A provides guidelines for implementing metadata over RTP. This implementation is based on an IETF (Internet Engineering Task Force) draft currently in application [20]. Example software code is available for download from the MISB web site.

4.1 RTP FMV Features

RTP has been designed to accommodate the nuances of internet-centric multimedia streaming. It offers the following capabilities:

- Delivery of digital motion video over various network and link types that may exhibit packet loss, packet re-ordering, latency, and jitter.
- A standardized method for requesting an RTP stream from a digital motion video provider.
- A standardized method for stream control to allow trick play.
- Authentication and encryption of data to provide integrity, confidentiality and non-repudiation.
- Provisions for lowering the overhead associated with packetizing and streaming data.

4.2 Relationship with established Internet Standards

The Internet provides a good example of the challenges faced when delivering data over disparate best-effort networks of varying qualities. ISMA [1] defines a set of standards for storage and streaming of media over the Internet; this RP aligns itself with that family of standards.

4.3 Relationship with established MISB Standards

The following figure illustrates the relationships between the current MISP standards and those in this document.

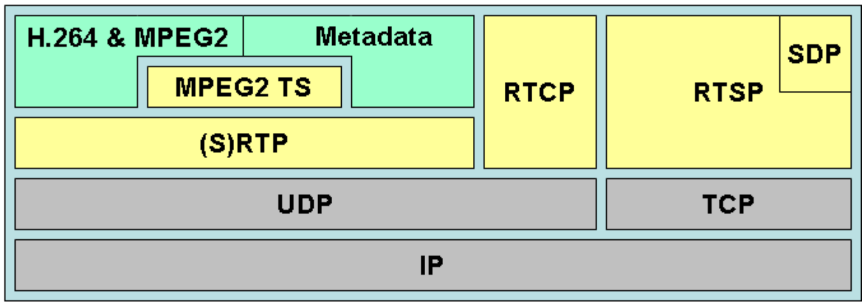

Figure 1 - RTP Relationship with Current MISP Standards (colored for visual aid)

Low-bandwidth digital motion imagery is intended to align with established MISP standards where appropriate; however, more optimal protocols are emerging for Internet use. Evaluation and adoption of new standards for more robust delivery is encouraged as part of the regular activity of the MISB.

4.4 Transport

The Real-time Transport Protocol is used to transport (near) real-time digital motion video. RTP provides end-to-end transport for media with real-time characteristics and is widely used in Internet streaming media applications.

The following RFCs provide the core RTP specifications that MUST be implemented. Payload specifications are detailed in following sections.

- RFC 3550 - *RTP: A Transport Protocol for Real-Time Applications* [2]
- RFC 3551 - *RTP Profile for Audio and Video Conferences with Minimal Control* [3]

Interleaved RTSP and RTP/AVP over TCP is an OPTIONAL method for transport. This method offers reliable transmission and more easily traverses Network Address Translation (NAT) devices and Firewalls at the expense of real-time time-critical response.

5 Data Formats

To provide broad applicability for devices that may only need one media component of the available data a client **SHALL** support both video and metadata delivery and MAY support audio. **This RP will address video and metadata only**.

5.1 Video

The **REQUIRED** codecs in this document are MPEG2 [4] and MPEG4 Part 10 H.264/AVC [5]. Coding parameters for H.264/AVC are defined in corresponding MISB Engineering Guideline EG 0802, *H.264/AVC Coding and Multiplexing* [6].

5.2 Metadata

The approved metadata sets are set forth in MISB EG 0104 [7], MISB STANDARD 0601 [8], and MISB STANDARD 0902[9] EG0104 specifies a mapping for Cursor on Target (CoT) data, which can be mapped for use in STANDARD 0601. Metadata that is time synchronized to the video is preferred over asynchronous methods. Time stamping of H.264/AVC video is described in MISB STANDARD 0604 [10]. RTP has provisions to carry a local timestamp for each media stream it carries, so although the video and the metadata will maintain independent timestamps the two streams can still be realigned at the decoder.

Metadata consistent with these Engineering Guidelines is encoded using the KLV (Key, Length, Value) construct according to SMPTE 336M-2001 [11].

The MISB-approved metadata structures include STANDARD 0902, *Motion Imagery Sensor Minimum Metadata Set* [9], STANDARD 0601, *UAS Datalink Local Metadata Set* [8], and RP 0701, *Common Metadata System: Structure* [12].

6 Native RTP Carriage of MPEG2 and H.264

MPEG2 native carriage over RTP specified by the IETF in [13] and by the DVB-IPI group in [14], and H.264/AVC native carriage over RTP specified in [15] with additional guiding parameter elections in EG 0802 [6] can be used with the following restrictions:

- The interleaved mode (packetization-mode=2) **SHALL NOT** be used, while the interleaved modes (packetization-mode=0, 1) **SHALL** be supported (guarantees lower latency).
- The parameters that are defined for interleaved mode (packetization-mode=2) **SHALL NOT** be present in the "a=fmtp" line in the SDP.
- The parameters max-mbps, max-fs, max-dpb, max-br **SHALL NOT** be present in the "a = fmtp" line of the SDP. These parameters extend the capabilities of a particular level specified by profile-level-id, but not all receivers may be able to decode streams beyond the profile-leve;-id specified.
- The format parameters line ("a=fmtp") in the SDP **SHALL** include the following parameters: sprop-parameters-sets, profile-level-id
- It is **RECOMMENDED** that the Sequence Parameter Set and Picture Parameter Set defined in [5] be carried in-band within the elementary stream. This should convey the same information as represented by sprop-parameter-sets.
- The SDP data **MUST NOT** change within a stream. If such a change is warranted a new SDP is issued and the stream restarted after the SDP data is communicated to the client.

7 RTP Carriage of MPEG2 Transport Stream

The RFC2250 [13] mapping shall be used as it provides a suitable mapping for MPEG-2 Transport Streams with further restrictions on RFC3550 and RFC2250 specified in [18] and [19]. Transport Stream over RTP incurs more overhead and increases the stream bit rate. There are benefits in transporting MPEG-2 TS over RTP because the packet count and time stamp in the RTP header offer a receiver the ability to detect lost and out-of-order packets, and in quantifying the jitter in packets received. Transport Stream over RTP may also be useful to transition from legacy tools, or to connect between points where such protocols may be difficult to modify.

8 RTP Carriage of Metadata

The MISB has submitted a draft IETF submission for a RTP Profile for KLV Encoded Metadata [20]. Reference to this document is made for native carriage of metadata. Appendix A further illustrates an implementation of this draft.

9 Native RTP Streams from MPEG2 TS Elementary Streams

MPEG2 transport stream allows a number of elementary media streams to be multiplexed together and carried as a unified synchronized package. This RP limits the media types that can be demultiplexed from a transport stream and produce individual RTP streams to video (MPEG2 and H.264) and metadata. While there is no reason that other media types multiplexed in a

MPEG2 TS stream, for example audio, cannot be similarily produced as an RTP stream RTP, at this time only video and metadata are considered.

Component media streams within a MPEG2 transport stream are synchronized together through the presentation time stamp (PTS) and decode time stamp (DTS) that accompanies each component (video, audio, metadata) packetized elementary stream (PES). The DTS directs the decoder when to decode the Presentation Unit (video picture, audio frames, etc.) of a particular media stream, while the PTS indicates when the decoded Presentation Unit is to be passed to the output device for display. The PTS thus provides a suitable timing reference that can be mapped as the timing reference for a corresponding RTP stream.

Guidance for mapping a video packetized stream to RTP can be found in [13] with mapping of the PTS to RTP timestamp stated as:

> *"Presentation Time Stamps (PTS) of 32 bits with accuracy of 90 kHz shall be carried in the fixed RTP header. All packets that make up a [video] frame shall have the same time stamp."*

Note: metadata will be mapped in a similar fashion referencing its corresponding PTS for synchronous carriage of metadata (see MISB STANDARD 0604 *Time Stamping Compressed Motion Imagery* [10]).

When a transport stream contains two or more component media streams that are to be produced as two or more RTP streams, for example, a video elementary stream and a private data stream (metadata), the Real Time Control Protocol (RTCP) should be used to support synchronization between the two media streams. RTCP provides a common reference clock (wall clock) shared by the individual media streams, and thus provides for resynchronization of the component streams at the decoder. Section 10.1 more fully describes RTCP.

10 Supporting Streaming Components

At a high level a motion imagery architecture consists of a data provider or source of motion imagery plus audio, metadata, etc. media components, and a data receiver of one or more of these same media components. These media components are typically synchrozied to one another; that is, the events occuring within a media correlate directly with events in the other media components. Lip sync is an example, where the mouth movements should correspond with the audio words spoken. RTCP (Real Time Control Protocol) is used to provide this synchronization between media components.

Stream playback control for play, stop, rewind, and fast forward is provided by a second protocol called RTSP (Real Time Streaming Protocol). Application needs will determine if either or both of these supporting protocols are warranted. Simpler configurations for single media communication with no control for instance—video or metadata only streams—can be done using RTP alone. In these cases, RTP's value is providing time stamp and packet count information useful in compensating for lost and re-ordered packets and network jitter.

The term “client” is used to refer to the endpoint receiving data from some data producer; this can be an end user (e.g., Warfighter) or another system.

10.1 Real Time Control Protocol (RTCP)

RTCP is an extremely useful companion protocol that provides bi-directional feedback between the sender and the receiver regarding the quality of a RTP session. As an example, RTCP allows a sender to provide a receiver a timing reference and how many bytes and packets have been sent. It allows a receiver to inform a sender about the quantity of packets lost, and a measure of the packet arrival jitter. RTCP usually accompanies streams delivered using RTP as a data carrier, and thus industry equipment may not function without it. At a minimum, servers **SHALL** emit RTCP sender reports (SR).

RTP time stamps, present in the RTP header, represent the sampling instant of the first octet of data in a media frame (video frame for example). The method to synchronize content transported in RTP is described RFC 3550 [2]. A simplified summary given for the synchronization of video and metadata is given below:

1. The RTP time stamp is expressed in units of a clock, which is required to increase monotonically and linearly. The frequency of this clock is specified for each payload format, either explicitly or by default. Often, but not necessarily, this clock is the sampling clock.
2. RTCP data is carried in RTCP packets. There are five types of RTCP packets, one of which is the Sender Report (SR) RTCP packet type. Each RTCP SR packet contains an RTP time stamp and an NTP time stamp; both time stamps correspond to the same instant in time. However, the RTP time stamp is expressed in the same units as RTP time stamps in data packets, while the NTP time stamp is expressed in "wallclock" time; see clause 4 of RFC 3550 [2].
3. Synchronized playback of streams is only possible if the streams use the same wall-clock to encode NTP values in SR packets. If the same wall-clock is used, receivers can achieve synchronization by using the correspondence between RTP and NTP time stamps. To synchronize a video stream and a metadata stream, one needs to receive an RTCP SR packet relating to the video stream, and an RTCP SR packet relating to the metadata stream. These SR packets provide a pair of NTP timestamps and their corresponding RTP timestamps that is used to align the media.

The update rate of RTCP sender packets is typically 5 sec, which means that upon entering a streaming session there may be an initial delay—on average a 2.5 sec duration if the default RTCP timing rules are used—when the receiver does not yet have the necessary information to perform inter-stream synchronization.

10.2 Control

The Real Time Streaming Protocol (RTSP) provides an application level protocol to interactively control the delivery of FMV digital motion imagery delivered via streaming. RTSP is defined in the following specification:

- Real Time Streaming Protocol (RTSP) [16]

RTSP provides a flexible protocol framework for controlling data streams with real-time properties. The following restrictions apply to ensure interoperability between endpoints supporting FMV data streams when this level of control is required or desired:

REQUIRED

- RTSP clients and servers **SHALL** implement all required features of the minimal RTSP implementation described in Appendix D of RFC 2326.
- RTSP clients and servers **SHALL** implement the PLAY method.
- RTSP clients and servers **SHALL** support RTP/AVP transport in the “Transport” header. When the RTP/AVP transport is used for a unicast session, clients **SHOULD** include the “client_port” parameter in the “Transport” header and servers **SHOULD** include the “server_port”, “source”, and “ssrc” parameters in the “Transport” header.
- RTSP servers **SHALL** send the “RTP-Info” header for unicast sessions.
- RTSP servers and clients **SHALL** support aggregated control of presentations.
- At most one RTSP session **SHALL** be “active” on a connection between an RTSP client and an RTSP server at any one time. An RTSP session becomes “active” when it is first referenced in a “Session” header. An RTSP session is no longer “active” after a TEARDOWN request has been issued for that session.

RECOMMENDED

- RTSP clients and servers **SHOULD** implement the DESCRIBE method. If the DESCRIBE method is implemented, it is **REQUIRED** that SDP be supported as the description format, as specified in Appendix C of RFC 2326[16].
- RTSP clients **SHOULD** generate the following RTSP headers when appropriate: “Bandwidth”, “Cache-Control”, “If-Modified-Since”, “User-Agent”. RTSP servers **SHOULD** correctly interpret these headers when present.
- RTSP servers **SHOULD** generate the following RTSP headers when appropriate: “Cache-Control”, “Expires”, “Last-Modified”, “Server”. RTSP clients **SHOULD** correctly interpret these headers when present.

10.3 Description and Addressing

Common to all setup and announcement protocols is the need for a means of describing the session. One commonly used protocol is the Session Description Protocol (SDP), although other mechanisms such as the Session Announcement Protocol (SAP) may be used. SDP provides a flexible text-based language for describing media streams and relating them temporally. SDP data **SHALL** be issued via FTP, SAP, RTSP, or HTTP. SDP is defined in the following specification:

SDP: Session Description Protocol [17]

When present, the SDP data MUST be formatted according to Appendix C of RTSP (RFC 2326[16]) at all times. Although Appendix C provides compatibility when delivering an SDP that will have media controlled via RTSP it allows for consistent formatting and attribute parsing.

11 Sample RTSP/RTP Session

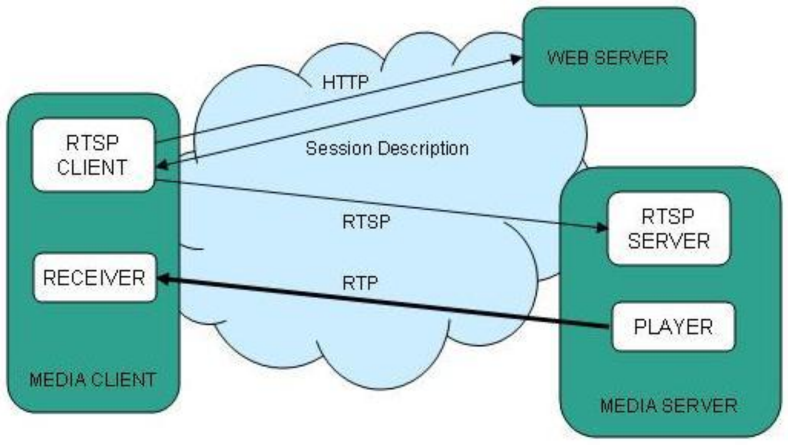

Figure 2 - RTSP/RTP Server/Client Communication

The following commands may be used to control a RTP session and are illustrated as state and requests in Figure 3:

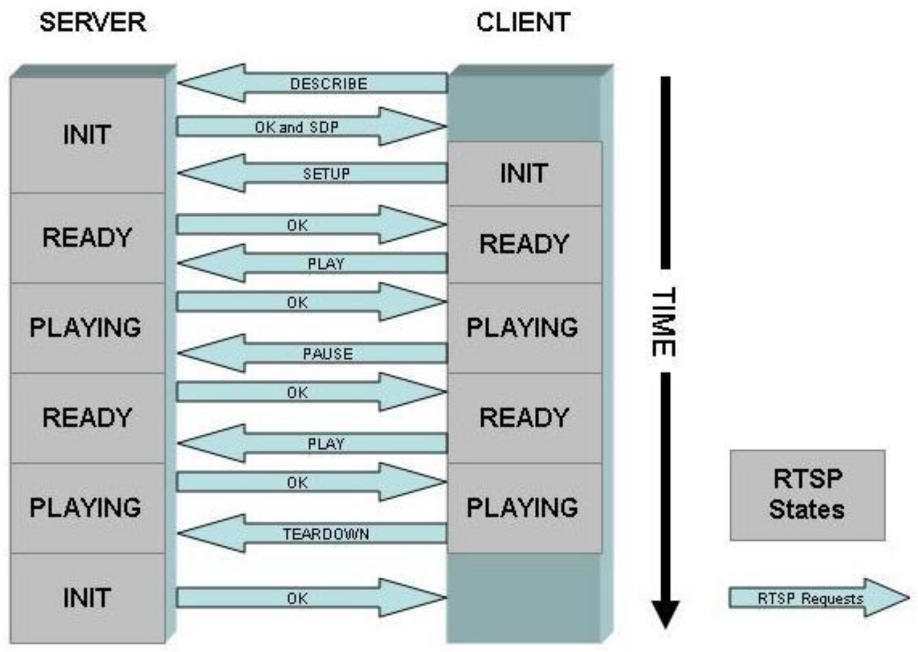

Figure 3 - Server/Client RTSP Interaction

DESCRIBE: Used by the client to retrieve a description of a presentation or media object on the server, corresponding to the *Universal Resource Locator* (URL) sent in the request. The response is typically in the form of the *Session Description Protocol* (SDP) and gives details such as the encoding types of the media, media duration, authors, etc. This command allows clients to find out more about a clip prior to streaming and also to check if the client can support the media format.

OPTIONS: Informs the sender what other valid requests it may issue i.e. what requests are supported by the corresponding client or server for the specified content at a given time. Illegal requests by either the client or server can be avoided with this operation.

SETUP: Transport of the requested media is configured using this command. Details such as the transport protocol and the port numbers to use are submitted to the server so the content is delivered in a manner appropriate for the client.

PLAY: Tells the server to start transmitting the requested media content as specified in the associated SETUP command. Unless a SETUP request has been issued and acknowledged, it is illegal to call the PLAY command. The absolute playback time and duration are also specified in this command so operation similar to fast forward and rewind on a VCR can be achieved with this command if the media can support such functionality e.g. live video streams cannot be scanned ahead.

PAUSE: Temporarily interrupts the delivery of the media currently playing in the session. PLAY must have been successfully requested in order to allow pausing of a video stream to a client. Resuming a paused media session is achieved using the PLAY request.

TEARDOWN: Terminates a stream or session. Once this command has been successfully issued the SETUP request must be called again before media content can be streamed again.

Other optional requests defined in the RTSP standard include ANNOUNCE, SET_PARAMETER, GET_PARAMETER, RECORD and REDIRECT. States for each session are maintained by the server to ensure that only valid requests are processed and that an error response is replied to invalid requests. To aid the server in determining if a request is valid a number of possible server states are used:

1. **Init:** the initial state meaning that the server has received no valid SETUP request.
2. **Ready:** the previous SETUP request was successful and acknowledged and the server is waiting to start or finish playing or a valid PAUSE request has been called.
3. **Playing:** a previous PLAY request was successful and content is currently being streamed to the client.

Figure 3 illustrates an example RTSP interaction between a client and server, highlighting the client and server states in response to different RTSP requests. In the session shown, the clients asks for a description of some content contained on the server using the DESCRIBE command and the server delivers information in SDP format with relevant details of the media queried by the client. The client then issues a SETUP request and the server configures the delivery of the stream to suit the clients preferred setup. PLAY is then sent by the client and the server starts transmitting the media data. A PAUSE request then prompts the server to halt the stream temporarily and the server waits in the Ready state for the client issue further instructions. Eventually the client requests PLAY and stream resumes. Finally, the client sends a TEARDOWN request and the server terminates the session and returns to the Init state waiting for other sessions to be established.

Revision History	
Date	
March 25, 2008	Initial Draft of document; approved for public release November 2008
March 17, 2009	Added MP2TS over RTP; producing RTP streams from MPT2S

Version 1	multiplexed components
May 09, 2009	Added SMPTE 2022 and Pro-MPEG references; updated references
May 20, 2010 Version 2	Added Appendix A reference implementation for metadata
Sep 30, 2010 Version 3	Language to: enforce SDP data need be communicated to client(s); clarify RTCP as normally required; clarify packetization-mode requirements; enforce SDP data remains fixed per session. Updated references.

Appendix A RTP Payload Format for SMPTE 336M Encoded Data Implementation Guidance (Informative)

A.1 Background

This documents a proof-of-concept implementation based on the IETF draft submission *RTP Payload Format for SMPTE 336M Encoded Data* [20], and is intended to serve as guidance for future "production class" implementations.

A.2 Definitions

KLVunit	a logical collection of all KLV items that are to be presented at a specific time. A KLVunit is comprised of one or more KLV items. Compound items (sets, packs) are allowed as per [SMPTE336M], but the contents of a compound item MUST NOT be split across two KLVunits. Multiple KLV items in a KLVunit occur one after another with no padding or stuffing between items.

A.3 Introduction

A client and a server implementation based on [20] are described in the following sections. To permit rapid prototype development, both implementations utilized and extended existing software applications. The guidance herein will be most directly applicable to implementers seeking to expand an existing RTP-capable software package with KLV/SMPTE336M carriage capability in accordance with [20]. However, completely new RTP implementations may also benefit.

For ease of session creation and association of multiple, related RTP sessions (video, audio, and KLV metadata) RTSP was used on both the client and server side of the implementation. RTSP provides a means for a client to connect to a server, ask for a description of available RTP-accessible media programs, and request that the server stream the media to the client over negotiated lower-level transport protocols (generally, UDP over IP).

It is assumed that the reader is familiar with general RTP concepts. Such background material can be found in [2,3,11,31].

A.4 Client Implementation

A.4.1 Basis for Implementation

The client implementation was constructed by extending an existing software package already capable of initiating RTSP sessions, synchronizing, and playing back multiple RTP-based sessions. Additionally, the software package could support SMPTE336M (KLV) parsing and interpretation of relevant KLV items from non-RTP sources (MPEG2 TS files or network streams, generally).

Because of the existing RTSP and RTP stack on the client end, as well as the capable KLV parsing and interpretation, implementation work was isolated to two main components: 1) extending the RTSP session initiation to recognize the KLV payload format in session descriptions (SDP), and 2) handling the new KLV payload format at the individual RTP session level.

A.4.2 Extension of RTSP Session Initiation

The client application already contained a table of known payload format type descriptions, keyed on media type name (legal media type names can be found in the IANA RTP Parameters Registry, http://www.iana.org/assignments/rtp-parameters). This table is used during RTSP session initiation when parsing SDP descriptions of the RTP sessions available from the server.

Entries in the table map the media type name to several processes needed for that type of media such as:

- Procedures to parse parameters in the SDP description that further describe the media's format
- Procedures to handle RTP packet headers and payloads consistent with the media format
- Identifiers that indicate how the raw media data is processed once the RTP layer processing is completed

Expanding the RTSP session initiation code to handle the new KLV payload format involved simply adding a table entry for the corresponding media type described in the IETF draft [20] ("application/smpte336m"). The entry provides the above three pieces of information as detailed below:

- **Parsing Payload Format Parameters:** The KLV payload format contains only one parameter, the clock rate. This is a very standard RTP parameter common to many payload formats. As such, the software already contained a re-usable routine to parse the rate parameter and subsequently use it to interpret RTP packet timestamps in the KLV-over-RTP session.
- **Procedures to handle RTP Packet Headers and Payloads:** Because of the similarities between the KLV/SMPTE 336M payload format specification and other popular payload formats, the implementation was able to re-use existing header and payload handling functions for the new KLV payload format. The next section describes the similarities and processing in further detail.
- **Identifiers Indicating How to Process Raw Media:** The software package already contained a KLV parser and processing framework; because of this, processing of KLVunits was simply mapped and routed to this pre-existing framework. See the next section for additional details.

A.4.3 Handling of the KLV Payload Format at the RTP Session Level

Once RTSP negotiation is complete, the individual RTP sessions are established. This section discusses the routines (mentioned briefly in the previous section) which handle the KLV/SMPTE 336M payload format.

RTP packet header and payload handling for the KLV payload format is based upon other popular video payload formats. A KLVunit is the same as a video frame in concept: it is all the data associated with a particular time, potentially spread across multiple RTP packet payloads for transport efficiency. The handling of the **M**-bit in the RTP header also parallels that of popular frame-based video formats. In addition, the KLV payload format does not define any additional RTP or payload headers. Because of the similarity with other popular video formats already supported by the software package, existing routines can be used without modification to provide proper RTP packet processing. The effective outputs of these routines are KLVunits together with their associated timestamps.

Once stripped of the RTP headers and concatenated as appropriate by the above-described handling mechanisms into whole KLVunits, processing of the KLV data is precisely equivalent to existing KLV processing and processing used with file or raw UDP network-based media. There is little concern about recovery from lost packets and data given the robustness of the existing KLV parser and processor. This downstream processor can deal with damaged KLVunits similarly to corrupt or malformed portions of file-based KLV streams. For implementations with error-intolerant KLV parsers, an additional step of discarding damaged KLVunits may be necessary to prevent problems in subsequent processing.

A.5 Server Implementation

A.5.1 Basis for Implementation

The server implementation was built by modifying the open source application "VLC" by the VideoLAN Project. VLC contains an RTP stack for streaming many popular video and audio formats from file-based media or re-streaming from other network sources. It also contains an RTSP server for session description and setup. This made it an ideal candidate for implementing the SMPTE336M/KLV payload format for RTP through extension.

To provide a concrete example for the community, as well as maintaining consistency with the open source philosophy that VLC is built upon, the actual source code modifications made to implement the KLV payload format for RTP is available for download from the MISB website.

A.5.2 Server Considerations

The server implementation was scoped to implement the KLV payload format for RTP described in [20] as a primary goal. To facilitate this, a data source had to be considered to provide the data for streaming. VLC presently does not process KLV metadata, so consideration went into support of various source data formats to avoid turning the RTP-oriented proof-of-concept into the much larger project of generically supporting KLV streams in VLC.

Based on this, and the broad availability of test data, it was decided to support MPEG-2 transport stream sources (from any source, including file or network) containing asynchronous KLV in private data streams (in accordance with SMPTE RP217) as the source for data to stream. VLC's transport stream demultiplexer needed to be modified to recognize and support KLV data.

Although MPEG-2 transport streams containing synchronous metadata would have proved a more accurate data source, asynchronous data sources were chosen because of the wide-spread availability of data samples.

A.5.3 MPEG-2 Transport Stream Demultiplexer Modifications

As mentioned above, VLC does not contain any notion of KLV metadata processing. While a complete description of the modifications necessary to provide this support is out of scope for this document, a brief discussion is provided for the purpose of conveying background context to the available download of the overall KLV payload format for RTP server-side implementation.

Routines were added to detect the registration format descriptor ("KLVA") in private data stream descriptions that indicate a SMPTE RP217 compliant KLV private data stream.
VLC does not contain a track "category" for metadata. Rather than complicating this rapid-prototype project with adding a new track category, it was decided to use the subtitles track category. Any detected KLV streams were mapped to subtitle-category tracks with the FourCC codec code "klva".

Routines were also added to the transport stream demulitiplexer to synthesize timestamps (based on the best-approximation stream proximity) of the otherwise timestamp-less asynchronous KLV data. The timestamps are used for further downstream delivery timing, and eventually used to populate RTP packet header timestamps.

A.5.4 KLV Stream-Out Packetizer

VLC's main streaming module (regardless of streaming protocol) is the "stream output" module. Prior to media being streamed, it is sent through a "packetizer" module. The packetizer creates basic payload units used by the streaming modules.

In the KLV payload format, the basic payload unit is the "KLVunit". As described in the client implementation above, the KLVunit is conceptually equivalent to a "frame" for video in that it contains all the data relevant to a given instant in time. Thus, a packetizer for KLV was authored to collect and concatenate any disjoint KLV data with the same timestamp.

This module ensures that the RTP output module receives whole KLVunits for output processing, thus completely taking care of the preparation for insertion into RTP packet payloads.

A.5.5 RTSP SDP Output

An RTSP client requests a description of the available RTP sessions when it connects to an RTSP server. To properly describe an available KLV session in accordance with the SMPTE336M/KLV payload format specification, an addition to the SDP synthesis code was added to map VLC streams with an internal codec identifier of "klva" to a media type code of "application/smpte336m".

The only payload format parameter for the KLV payload format is the clock rate; because this is such a common parameter, VLC already defaults this appropriately for all RTP stream descriptions.

A.5.6 RTP Packet Header and Payload Creation

The KLV payload format is very similar in its use of the RTP header (timestamp, M-bit) to most popular video payload formats (this is described in further detail in the client implementation guidance above). Because of this similarity, and the fact that VLC already supports RTP streaming of many popular video formats, it was possible to reuse existing packet creation functionality.

The existing packet creation function, intended primarily to take video frames and create one or more RTP packets, could be directly applied to a KLVunit.

It is expected that most existing, robust RTP implementations can be extended to support the KLV payload format by reusing existing functionality in this way.